In *Proceedings of the 2nd NASA International Workshop on Planning and Scheduling in Space*. San Fransisco, CA. 2000

Reactive prioritization

Michael Freed
MS 262-4 NASA Ames Research Center
Moffett Field, CA 94035

mfreed@mail.arc.nasa.gov

Abstract

In many interesting task environments, agents must decide priority among competing tasks under considerable uncertainty. Moreover, which kinds of priority-relevant information are available for a decision will vary in different situations. An ideal priority determination process should use whatever information is available, even it becomes so just before a decision is required or after a task has been awarded priority and begun executing. In this paper, we identify several kinds of priority-relevant of information and describe a flexible priority computation method that uses whatever kinds are available in a given situation.

Introduction

In everyday environments such as the kitchen or automobile, people usually have many things they could reasonably be doing at a given moment. Some of these tasks are independent and can be pursued concurrently. Others interact in ways that demand a choice: which should be given priority and carried out immediately? Which should be deferred, interrupted, or aborted?

For artificial agents, one approach to making such decisions is to identify all tasks to be carried out and all the constraints on those tasks, then search for the best possible order. While this can produce optimal or nearly optimal action orderings, it is only practical in highly predictable environments where needed actions and relevant constraints are known in advance. Many everyday environments are not predictable in this sense. Unplanned actions may be needed to handle unexpected events – e.g. a car suddenly cuts in just ahead, the phone rings, an awful song comes on the radio. Similarly, the timing and specific actions needed to carry out a task may not may not be known until it is nearly time to carry them out. For example, an agent may know that it will have to turn left onto Elm Street, but not know which particular driving maneuvers will be needed to negotiate traffic or whether it will first have to stop at a red traffic signal.

An alternative approach, *reactive prioritization*, is to make rapid priority decisions just before committing to a course of action. Unlike the more deliberative approach in which priority decisions are made arbitrarily far in advance of execution, a reactive prioritization process makes such decisions in response to newly available information about, e.g., which tasks are eligible for execution at a given moment, whether they interact, and what timing constraints apply to each. While these kinds of information are sometimes available far in advance, they often remain uncertain until the last moments before a task becomes enabled.

Pervasive, priority-relevant uncertainty has at least two important implications. The first, as discussed, is that no maximally-informed priority decision for a task is possible until that task is eligible for execution. Thus, priority must be determined at execution-time, not (only) in the course of long-term planning. The second is that priority decisions will often have to be made in the absence of potentially significant information. For instance, not knowing about an impending task to answer the phone makes it impossible to choose between tasks on the basis of which better tolerates interruption. The best an agent can do is to use whatever information is available when a final (or otherwise consequential) priority decision is needed.

In designing reactive priority mechanisms, it is important to consider what types of information are likely to be available at decision-time and how each type should influence task priority. In this paper, we identify several kinds of information that may be available, each of which can be used as a heuristic basis for deciding priority. These heuristics are combined into a more general prioritization process that takes advantage of situations in which more than one kind of priority-relevant information is available. This paper extends work reported in Freed (1998) and has been implemented as part of a plan execution system used to simulate expert human operators doing complex tasks (Freed and Remington, 1997).

Heuristic Prioritization

Tasks need to be prioritized when they conflict. For example, when two tasks require looking at widely separated points in the visual field, prioritization is needed

to decide which gets to control the orientation of visual sensors. In some cases, not getting priority merely implies that a task will have to wait (or be interrupted if it is already ongoing). In others, the task cannot wait; deferring it means causing it to fail. Deferring a task imposes a (possibly zero) expected cost. More specifically, each task can be viewed as having a set of deadlines, each of which incurs a specific expected cost if missed; deferring a task increases the likelihood that one or more of its deadlines will be missed. For instance, the task of getting home from work might involve deadlines for arriving on time for a planned dinner, seeing a televised sports event from the beginning, and having time with the kids before they go to bed. The longer the task of driving home is deferred, the greater the risk of "deadline bust" to each associated deadline. *The job of an agent's priority mechanisms is to minimize the total expected cost of deadline busts across all tasks over an extended period.*

Designing a prioritization mechanism able to do this job in an uncertain task environment requires first identifying kinds of priority-relevant information that might be available at decision-time. In the worst case, the agent only knows the identity of tasks currently being considered for execution. In this situation, there is no choice but to select randomly between competing tasks. A somewhat more benign case occurs if there is information on one priority-relevant task attribute. These attributes include anything that affects the likelihood of missing a task deadline including: (1) the time remaining until a particular deadline bust, (2) the cost of missing a deadline, (3) task duration, and (4) the time cost resulting from task interruption. The role of each of these attributes in determining priority can be expressed as a set of heuristic decision rules as follows (cf. Firby, 1989):

All else being equal, …
do the task with the nearer deadline first
do the task with the most important deadlines first
do the briefer task first
do (continue) the ongoing task rather than switch

These rules make it possible to decide priority when only one kind of information is available. Ideally, a prioritization mechanism will make use of more information if more is available. To understand how such a mechanism should be realized, it is worth considering each of these information types in greater depth.

Urgency. Prioritization is only an issue because there may be undesirable consequences to deferring a task. The idea of a deadline, after which undesirable consequences will occur, is thus likely to be a central feature of any approach to prioritization. We generalize this idea slightly by treating deadline-pressure as one (very important) source of *urgency*. Others possible sources include value-decay-pressure in which the cost of waiting comes gradually rather than all at once (e.g. dinner slowly cools down and becomes less enjoyable to eat), and hazard-pressure where the likelihood of an undesirable occurrence remains constant for a given interval rather than approaching certainty during a given interval (e.g. lightbulb goes out). These forms of urgency differ in the shape of the expected cost curve. For a deadline, the curve remains relatively flat for a while and then quickly approaches maximum. A hazard function smoothly asymptotes to maximum following a curve of the form $1-b^t$ $\{0<b<1\}$. Value decay tends to increase linearly to maximum. However, in all forms of urgency, longer execution delay means greater expected cost.

Importance. Another crucial factor is the measure of how bad it would be to miss a given deadline. [1] This is especially relevant if meeting a deadline for one task precludes meeting a deadline associated with another. In such cases, the best thing to do is to make the more important deadline – i.e. avoid the effects of the bust with greater negative utility.

Duration. A task's duration affects its proper priority in two ways. First, doing one task requires deferring all competing tasks, thereby increasing the risk of deadline-busts for all these tasks' associated deadlines. Thus, as a task's duration increases, so does opportunity cost of carrying it out. The importance of this effect increases not only with increased task duration, but also with the number of tasks deferred. It is often desirable to execute a number of brief tasks before starting one of longer duration, even if the lengthier task is more important.

A second effect of duration is to reduce the amount of time available before a given task must be started (or resumed). For instance, a 2 minute duration task with a 5 minute completion deadline should be started within 3 minutes; doubling the duration but leaving the deadline constant would allow only one minute to start the task. This urgency-increase effect opposes the effect of increased opportunity cost since a longer duration task apparently needs to be started sooner.

The heuristic "all else being equal, do the briefer task first" applies because the opportunity cost effect is greater than the increased-urgency effect. If we assume that competing tasks have the same number of equally distant and equally costly deadlines, then the only basis for comparison is the amount of increased risk (a monotonically increasing function of deferral time) to

[1] Currently, all forms of urgency are treated analogously to deadline-pressure. Value-decay-pressure is handled by selecting some decay threshold to treat as a deadline-bust. Thus, one might assume that dinner will stay hot for 5 minutes; after this (arbitrary) deadline, it is assumed to have become "not hot" and thus a suffered cost. Hazard pressure is handled by selecting a probability threshold p and treating the time taken for an undesired event to have occurred with probability p as a deadline.

each. For example, assume there are five tasks to do, four with 1 minute durations and one with a 5 minute duration. Doing the latter task first imposes a combined 26 minutes of deferral on all tasks. Doing the task last imposes only 10 minutes. The next section discusses an approach to prioritization that accommodates both duration effects.

Interruption cost. Priority should be recomputed whenever changed circumstances indicate that a previous priority decision may have become obsolete. For example, when a task becomes enabled and thus eligible for execution, its priority should be assessed against any competing tasks including those ongoing. Similarly, situation changes that either lower the likely priority value of an ongoing task or increase the value of a previously deferred task should trigger reassessment of priority decisions. When reprioritization reveals that an ongoing task should have lower priority than a competitor, the task should be interrupted.

However, interruption imposes costs that should be considered when computing the priority of an ongoing task. For instance, interruption may require time-consuming transitional activities to avoid consequences of an overly-sudden interruption – e.g. pulling over to the side of the road to avoid crashing when interrupting a driving task. Other activities may be required to maintain task viability during the interruption interval and to correctly resume (Freed 1998). These activities require time and resources that may be useful for other tasks and thus impose an opportunity cost.

Interruption may impose other costs as well. Important preconditions may be undermined during the interruption, making the task impossible or more expensive. Facilitating conditions (opportunities) may lapse, raising task cost.

Priority mechanisms should take the expected cost of an interruption into account, suppressing interruptions unless the expected benefit exceeds the expected cost. When cost/benefit information is not available, the agent should assume a greater-than-zero interruption cost and inhibit interruption.

Robust Prioritization

The four heuristics listed above are useful for comparing candidate tasks, each on the basis of a single type of priority-relevant information. Ideally, an agent should take advantage when more information is available. One fairly simple case to consider is when the agent knows both the deadline proximity and duration of candidate tasks. Since deadline describes when the task should be completed, duration can be subtracted from this time value to determine when the task should be started. This is much more valuable measure of task urgency than deadline alone, since it makes it possible to determine how much time can be spent on alternative tasks before the risk of a deadline bust rises dramatically. Thus, as measured urgency (U = deadline-proximity – task-duration) approaches zero, priority should rise to task maximum.

The computation of urgency can be further refined by considering the effect of deferring a task in favor of some alternative. For example, if a task has a 5 minute completion deadline and lasts 3 minutes, then deferring it in favor of a competitor that also takes 3 minutes to carry out will cause a missed deadline. To avoid such scenarios, priority mechanisms should not only consider a task's duration, but also the duration of the strongest competing task. When the requisite information is available, priority should be determined on the basis of a tasks' *adjusted-urgency* (U' = U – competitor-duration).

A more complicated case is when information on both the time remaining until deadline (unadjusted urgency) and the cost of missing the deadline is available. What makes this complex is that the relevance of these two factor depends on another factor: how busy the agent is in the timeframe in which the tasks needs to be carried out. When the agent is not very busy and, thus, probably has time to do all intended tasks, the priority mechanism's job should be to make sure that a bad ordering decision doesn't lead to an unnecessarily missed deadline. In this case, urgency is a highly relevant determinant of priority while importance (the cost of a missed deadline) is largely irrelevant. In contrast, when the agent is so busy that there isn't time to do everything, missed deadlines are inevitable. In this case, the priority mechanism needs to make sure that important deadlines are met, even if that means allowing less important deadlines to bust. To handle both importance and urgency in a unified framework, priority mechanisms need to employ some measure of busyness.

To see how busyness might be usefully represented, it is worth formally characterizing the overall purpose of the prioritization process. A useful characterization, as indicated earlier, is that the process should try to minimize the long-term cumulative cost of missed deadlines. With respect to any given priority decision, the task whose deferral would add most to this total cost should be selected. A task's expected cost of deferral (ECD) is computed by summing the expected deferral cost for each of its associated deadlines. Thus,

$$\mathrm{Priority}(task) \propto ECD(task) \propto \sum_{d}^{deadlines} p(d) * Cost(d)$$

where *d* is a *task's* deadline, *p(d)* is the probability of missing *d* if *task* is deferred in favor of its current strongest competitor, and *C* is the cost of missing *d*. The identity and cost (importance) of the task's deadlines are assumed to be known. The probability of missing a given deadline

must be derived from how much time is available to do the task, U' (adjusted-urgency), and how great the task load will be (busyness) when the task, if deferred, again becomes eligible for execution. In particular, measured busyness must be a probability distribution B of minimum task delays – i.e. the probability that a deferred task will have to be further deferred for greater than or equal to a given time t. The value of p(d) is thus B(t) for given busyness distribution B with t = U'.

Consider a simple example. Task Q is eligible for execution. Its duration is 2 minutes. It has one associated deadline whose importance (the cost if the deadline is missed) is 7 and whose urgency (how much time is left to complete the task in order to avoid missing the deadline) is 5 minutes. Q is competing with other tasks for priority including the current strongest contender, R. R's duration is also 2 minutes. If R is selected for execution over Q, there will be (5-2-2) = 1 minute left in which to start Q after R is completed. This value is Q's adjusted-urgency. Given a busyness distribution B in which the value B(1 minute) = .3, .3 is the likelihood that, given all the other tasks that need doing, there will be at least a 1 minute delay before a given task can be started. The cost of deferring Q is thus .3*7 = 2.1. If this value is higher than that for R, Q becomes the strongest competitor in the priority competition.

Three problems remain to be solved to make this approach viable. The first and simplest one is to handle the possibility that needed information is not available – i.e. if a task's duration, deadline urgencies, or deadline importance values are unknown. The simplest approach is to factor out the influence of the missing information by assuming the same value as that of the task's strongest current competitor. For instance, if R in the example above has duration 2 minutes but Q's duration is unknown, assume a duration of 2 minutes. Similarly, if an unadjusted urgency value is missing, set it equal to the average of R's urgency values. If R's value is unknown, an arbitrary default value can be used instead.

The second problem is how to determine priority for an ongoing task. As mentioned earlier, interrupting an ongoing task may entail various costs such as the opportunity cost of time spent safely winding down (safing) the task and then, later, restoring any preconditions violated during the interruption interval. Our approach is to assume that the priority of an ongoing task is ECD + expected-interrupt-cost where the latter term is either known for the task or defaults to some nominal positive value. This has the effect of inhibiting interruptions in proportion to their undesirable consequences (Freed, 1998; cf. Gat, 1992).

The third problem is how to determine the shape of the minimum delay (busyness) distribution B. A few shape attributes are obvious. For example, p(0) = 1.0 – i.e. it is certain that a task will be delayed for at least 0 time. B is monotonically decreasing; thus, the more urgent the task, the greater likelihood of an unavoidable deadline bust. B asymptotes to 0 since, in principle, there is no limit to the amount of delay that might be required. Beyond this, B's distribution depends on various factors as described below:

Projection. In delaying a task for less important but more urgent alternatives, an agent risks losing the opportunity for timely execution because of emerging, higher priority tasks. To prevent this, the agent should take advantage of any predictions it can make about the set of future tasks a given current task will have to compete with if deferred. These future tasks can be divided into two categories: known and unknown. Known tasks are those that the agent already intends to execute when it gets a chance, including those whose enabling preconditions have not yet been satisfied. The unknowns include not-yet-specified subtasks of known tasks and tasks that do not yet exist (e.g. to handle a future phone call). Even the unknowns can be predicted to some degree. For example, an agent may know that this is a time of day when many phone calls should be expected, even if no specific answer-phone-call tasks currently exist. Similarly, it may be possible to project the most likely decomposition of a task into subtasks, even if the final decomposition decision has not been made.

The best possible delay estimate would, of course, be based on the most detailed and complete possible projection of future tasks. The described prioritization approach uses very coarse projections. Only a single known task, the strongest current competitor, is treated individually (used to compute adjusted-urgency). All other tasks including lower priority enabled tasks, non-enabled tasks, and unknown tasks, are considered in the aggregate via the busyness function. If instead, all known tasks were handled by explicit projection, the busyness function would only have to account for residual task-load – i.e. the "unknown" tasks.

Thus, the proper shape of this function depends on how projection is used in the prioritization process as a whole. Given the modest use of projection in our current approach, the function should incorporate any relevant, available information about future tasks. For example, if the total number or average duration of known tasks is relatively high, the distribution should be skewed towards relatively long delays. If an imminent surge of new (not yet existing) tasks is expected, expected delay should be proportionately greater.

Task attributes. Any delay distribution must make additional assumptions about the effect of task attributes – particularly duration and importance – on future priority decisions. Importance is significant because the more important a task is, the higher its priority will rise as the task deadline nears. This implies that importance should have two opposing effects on priority. Since the cost of

missing an important deadline is relatively great, importance increases expected-deferral-cost and, thus, priority. But because the likelihood of having to further delay an important task for even more important tasks is relatively low, the probability of a deadline bust is lower, making priority lower. The sum of these opposing effects should vary with urgency: as the deadline approaches, it is more crucial that the task be a strong competitor now rather than permit a delay that counts on it being a strong competitor later.

To the extent that short tasks have an advantage in gaining priority, a task's duration is also relevant for determining what kind of deferral distribution should be assumed. By raising a task's urgency in proportion to it's strongest competitor's duration, the prioritization process outlined above essentially penalizes long duration tasks – i.e. it has a strong short-duration bias. This bias should be taken into account in the expected-delay distribution by assuming relatively short delays for brief tasks.

Task duration affects B in other ways as well. First, longer tasks can often be interrupted for brief periods without significant effect. For example, a driver can look to the side of the road for a brief period without much risk, even though a longer look would significantly interfere with driving. An agent can take advantage a task's insensitivity to interruption to carry out brief tasks (Freed 1998). Similarly, lengthy tasks often have built in idle (slack) times during which brief tasks can be "fit in." For example, a driver stopped at a red light may have a brief time to use hands and eyes freely for non-driving tasks. The ability to execute brief tasks during idle intervals or within permissive interruption constraints means that short tasks can effectively be run concurrently with longer tasks they nominally conflict with. All of these effects imply that shorter tasks are less likely to be delayed for a long period.

Domain specificity. Another set of factors affecting the true distribution of task delays arises from characteristics of the task environment. This is especially the case for estimating the incidence of new tasks (e.g. rate of new phone calls, new customers at a bank, new aircraft for an air traffic controller,..). Environmental factors can also affect busyness less directly. For example, in conditions that make task failure likely, more time will have to ve spent on failure recovery. This essentially increases expected task duration and, thus, busyness.

An approach that makes it possible to address all of these factors is to assume a normal distribution of delays, using the complement of the cumulative distribution (a sigmoid) for B; each factor mentioned above affects the mean of the distribution. For instance, an increase in the expected incidence of new tasks increase the mean while a reduction in the average importance of known tasks reduces it. To account for the very significant effect of task duration D, we assume a default distributional mean equal to D. This captures the idea that a task is typically competing with a small number of tasks of about its own duration. Tasks that are significantly shorter will likely be executed earlier and with little effect on overall delay, while tasks that are much longer will be executed later or pseudo-concurrently.

Conclusion

In realistically uncertain worlds, an agent often lacks advance knowledge of what new tasks will arise and what specific actions will be needed to make progress at known tasks; thus, it cannot consider characteristics of these unspecified later actions when deciding order among earlier ones. Moreover, an agent often cannot be sure when an opportunity to execute its tasks will arise, and therefore cannot decide whether the task should come before or after some other, independently enabled task. With "when" and "what" uncertain, agents must instead reactively prioritize between currently eligible tasks based on whatever information is available.

This paper presents an approach based on the idea that the role of a prioritization process should be to minimize the cost of missed deadlines over a lengthy interval. The approach is designed to make best use of whatever priority-relevant is available at decision-time. If some useful piece of information is not available, the priority process should behave in a robust fashion, essentially falling back on simpler, more general decision heuristics. The greatest challenge has been to design an approach that flexibly decides whether to focus on meeting urgent deadlines or whether to insure that the most important deadlines are met.

The described approach captures a wide range of factors affecting priority decision-making under uncertainty. Furthermore, it is simple and computationally inexpensive enough to be realized in a practical agent architecture (Freed and Remington, 1997; Freed, 1998). However, the approach falls short of ideal in several of ways. First, contingent behaviors associated with a task such as dealing with failure, managing periodic behavior, coping with undesirable side-effects, are important contributors to task load but have yet been accounted for. Perhaps more importantly, the current approach makes too little use of projection The current approach can only make use of information on a single projected task (the one with the highest current priority), even though characteristics of other tasks may be known. Finally, the problem of setting the crucial mean-delay parameter is left almost entirely to domain-specific rules. While there may be no avoiding the need for such rules, the qualitative discussion of the effects of various factors does not provide a clear methodology for creating them. Future work should remedy these deficiencies.

References

Firby, R.J. 1989. Adaptive Execution in Complex Dynamic worlds. Ph.D. thesis, Yale University.

Freed, M. & Remington, R.W. 1997. Managing Decision Resources in Plan Execution. In Proceedings of the Fifteenth Joint Conference on Artificial Intelligence, Nagoya, Japan.

Freed, M. (1998b) Managing multiple tasks in complex, dynamic environments. In Proceedings of the 1998 National Conference on Artificial Intelligence. Madison, Wisconsin.

Gat, Erann. 1992. Integrating planning and reacting in heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of 1992 National Conference on Artificial Intelligence*.